



Nokia 3510i Content Authoring SDK

Version 1.0

User's Guide

February, 2003

Copyright © Nokia 2002-2003. All rights reserved.

This document is for use with the Nokia 3510i Content Authoring SDK. Reproduction, transfer, distribution or storage of part or all of the contents in this document in any form without the prior written permission of Nokia is prohibited.

Nokia, Nokia 3510i, and the Nokia Connecting People logo are trademarks or registered trademarks of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries.

Nokia operates a policy of on-going development. Nokia reserves the right to make changes and improvements to any of the products described in this document without prior notice.

UNDER NO CIRCUMSTANCES SHALL NOKIA BE RESPONSIBLE FOR ANY LOSS OF DATA OR INCOME OR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES HOWSOEVER CAUSED.

THE CONTENTS OF THIS DOCUMENT ARE PROVIDED "AS IS". EXCEPT AS REQUIRED BY APPLICABLE LAW, NO WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE MADE IN RELATION TO THE ACCURACY, RELIABILITY OR CONTENTS OF THIS DOCUMENT. NOKIA RESERVES THE RIGHT TO REVISE THIS DOCUMENT OR WITHDRAW IT AT ANY TIME WITHOUT PRIOR NOTICE.

The availability of particular phone products may vary by region. Please check with the Nokia dealer nearest to you.

Visit Forum Nokia (<http://www.forum.nokia.com>), the site designed for developers using technologies supported by Nokia.

Table of Contents

Nokia 3510i Content Authoring SDK

User's Guide

Preface

Document Overview	3
Audience	3
Typographical Conventions	3
Other Product Documents	3
Supporting Application User Documents	4
Related Forum Nokia Documents	4
Related WAP Documents	4

Introduction to 3510i Content Authoring SDK

What is Nokia 3510i Content Authoring SDK?	5
Configuring Nokia 3510i Content Authoring SDK	6
Using Nokia 3510i Content Authoring SDK With NMIT.....	7
Installing NMIT 3.1.....	7
Configuring NMIT.....	7
Configuring a Proxy Within NMIT	7
Using Nokia 3510i Content Authoring SDK With Adobe GoLive.....	8
Installing Adobe GoLive.....	8
Configuring Adobe GoLive for Previewing WML Content.....	8
Configuring Adobe GoLive With NDS for MMS Beta 0.3.....	9
Configuring Adobe GoLive With NDS for MMS 1.0	9

Browsing, Receiving, and Creating Messages

About Browsable Content Types	11
Memory and Storage Constraints for Browsing	12
Browsing in the Standalone Nokia 3510i Content Authoring SDK.....	12
Browsing Within NMIT.....	13
Browsing Within Adobe GoLive.....	13
Receiving MMS Messages	13
Overview of MMS Messages	14
Content Types Supported Within an MMS Message.....	15
Memory and Storage Constraints for MMS Messages	15
MMS Messages in the Network and Simulation Environments.....	15
Creating MMS Messages.....	16
Creating an MMS Message Using NMIT	16
Creating an MMS Message Using Adobe GoLive	17
Receiving MMS Messages.....	18
Receiving an MMS Message Using the SDK GUI.....	18
Receiving an MMS Message Using NMIT	18
Receiving an MMS Message Using Adobe GoLive	18
Viewing and Disposing of an MMS Message.....	19
Receiving Push Messages	20

Push Message Content Type Support	20
Viewing Push Messages Created in NMIT	20
Disposing of Push Messages	22
Browser Features	22
Rendering	22
Screen Attributes	22
HTTP Features Supported	22
WAP-Specific Functions Supported	22
Character Set Support	22

Using Nokia 3510i Content Authoring SDK

Menu Commands	25
File Menu	25
Tools Menu	26
Help Menu	28
Phone Menus	29
Running the SDK Through a Command Line Interface	29
Command Line Syntax	29
About Options	30
Options for Starting the SDK	30
Options for Managing SDK Instances	32
About Parameters	33
Working With a CLI Shell	33
Keyboard Shortcuts	34
QWERTY Text Input	35

Preface

Document Overview

This manual describes how to use the Nokia 3510i Content Authoring SDK. The Nokia 3510i Content Authoring SDK is an application that simulates the features of a Nokia 3510i mobile handset that are related to mobile Internet browsing and message (MMS and Service Indication) reception. Calling features of the mobile handset are not operative in the SDK, though the related menus and options are present.

Audience

The Nokia 3510i Content Authoring SDK is designed to accommodate the needs of developers of content for browsing, MMS, and Push applications intended to be shown on the Nokia 3510i mobile handset.

Typographical Conventions

The following typographical conventions are used in this guide:

Notation	Explanation
<code>Courier</code>	<ul style="list-style-type: none">• Text that you enter (as opposed to system prompts and responses)• File paths• Commands• Program code
<i>Italic</i>	<ul style="list-style-type: none">• Names of books and documents• New terminology
Bold	Names of Windows menus, commands, buttons, and icons
URL link	Active link to an external URL.

Other Product Documents

- *Nokia 3510i Content Authoring SDK, Release Notes*
- *Nokia 3510i CA SDK, Installation Guide*

Supporting Application User Documents

- *Nokia Mobile Internet Toolkit User's Guide, Version 3.1*
- *Adobe GoLive Installation Guide, Version 6.0*
- *Adobe GoLive User Guide, Version 6.0*
- *Nokia Developer's Suite for MMS Installation Guide*
- *Nokia Developer's Suite for MMS User's Guide*

Related Forum Nokia Documents

The following documents, in addition to many others, are created and maintained by Forum Nokia. They are freely available at <http://www.forum.nokia.com/>.

- *How to Create MMS Services*
- *Nokia in Messaging white paper*
- *MMS Conformance Document*
- *Nokia WAP Phone Characteristics*
- *WAP Service Developer's Guide for Nokia Series 30 Phones With WML Browser*

Related WAP Documents

The following specifications, in addition to several others, are created and maintained by WAP Forum. They are freely available at <http://www.openmobilealliance.org>.

- *WAP Wireless Markup Language (WML) Specification*
- *WAP WMLScript Language (WMLScript) Specification*
- *WAP WMLScript Standard Libraries Specification*
- *WAP Push Architectural Overview*
- *WAP Service Indication (SI) Specification*
- *WAP MMS Architecture Overview*
- *WAP MMS Client Transactions*
- *WAP MMS Encapsulation Protocol*

Introduction to 3510i Content Authoring SDK

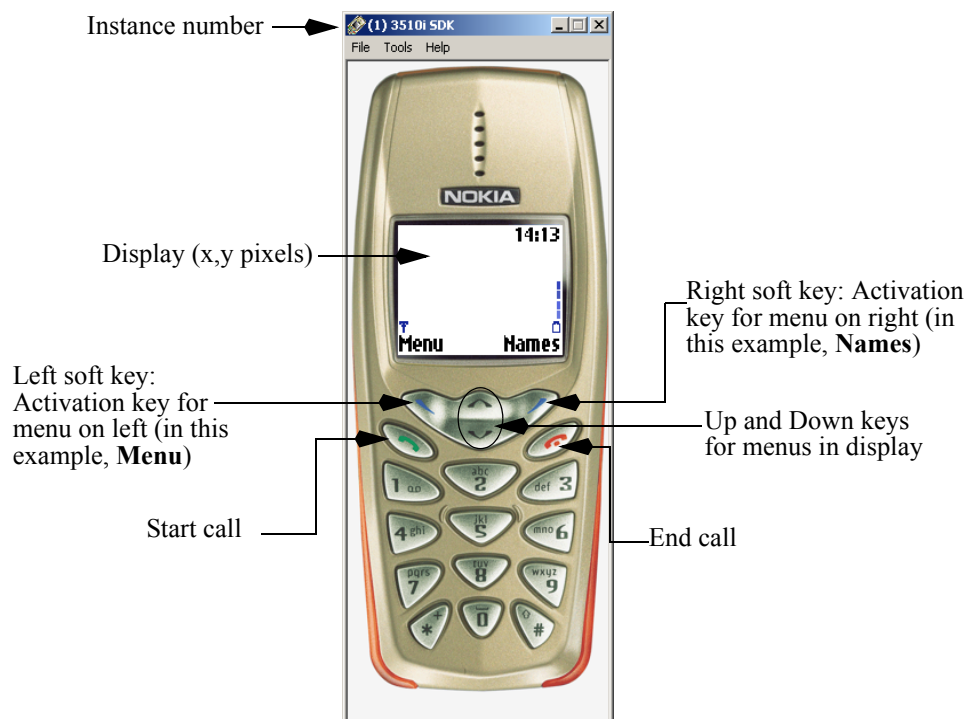
This chapter provides an overview of how you can use Nokia 3510i Content Authoring SDK as a standalone application or within a supporting application.

What is Nokia 3510i Content Authoring SDK?

Nokia 3510i Content Authoring SDK is a development tool for mobile Internet content developers who want to preview how their content will look when it is ultimately deployed on an actual Nokia 3510i mobile handset. The SDK is designed for previewing browsable WML content, Multimedia Message (MMS) reception, and Push message reception.

The SDK can be used as a standalone application or with two other supporting content development applications: Nokia Mobile Internet Toolkit (NMIT) and Adobe GoLive. NMIT or Adobe GoLive content developers can deploy their content to the SDK to test the content before deployment to the 3510i mobile handset.

The SDK's application window is displayed below. Mouse clicks on the phone buttons simulate pressing the keys on a 3510i mobile handset: these mouse clicks map to functions in the phone software which simulate the functions available in the actual mobile handset:



The SDK can load mobile Internet content in any of the following ways:

- Through a WAP gateway to a remote Internet server. This method is called *browsing*. To load this type of content, you must configure the SDK to use an external WAP gateway, or you must use the SDK together with NMIT, which uses a server simulator as its own WAP gateway. In either case, you must configure the SDK for this connection (**Tools>Settings>WAP networking**).
- By opening a WML file on the local network (**File>Open** and then **Browse** to locate the desired file).
- By loading content from a supporting application. This method is called *previewing*. Both the NMIT and Adobe GoLive enable direct display of content authored using their content development editors. The previewing mechanism is either a button within the editing window or a menu selection from the application menu bar.

Configuring Nokia 3510i Content Authoring SDK

This section describes what you need to know about installing supporting applications to use them with the SDK. However, it does not provide a step-by-step description of installation procedures. For this information, see the User or Installation Guides of the individual products.

After installing the SDK, you can configure its settings. When you run the SDK as a standalone, you access these settings through the **Tools>Settings** menu item. When you run the SDK from within NMIT, you access these settings through the NMIT **Settings>Device Settings>Other Settings**. Settings are identical.

The **WAP networking** tab in the **Settings** menu enables you to enter a WAP Gateway through which the SDK can connect to Internet content servers; this is required if you want to use the SDK to browse Internet content.

By default, the SDK is configured to use the WAP gateway at IP address 127.0.0.1. This is the IP address of Nokia's WAP server simulator, which is a single-user WAP gateway running on the user's computer. The SDK installation does not install this server simulator; so, if you want to use it as a WAP gateway, you must install NMIT (whose installation does include this simulator). Further, NMIT must be running when you use the SDK; otherwise, the server simulator would not be available (NMIT shuts it down upon exit).

If you access the Internet through a proxy (as you might, for example, if your computer is within a corporate Intranet) and if you want to use the server simulator as your WAP gateway, you must configure the server simulator (from within NMIT) to use the proxy server. See the section titled [Configuring NMIT](#) for additional information about this configuration.

When you configure the SDK to use NMIT's server simulator as its WAP gateway or to use an external gateway, you can browse both local files and mobile Internet content.

Using Nokia 3510i Content Authoring SDK With NMIT

NMIT 3.1 is available for downloading from the [Forum Nokia web site](#). Using NMIT with the SDK enables you to explore the Push and MMS messaging features of the SDK; in addition, it provides increased browsing and debugging capability.

Installing NMIT 3.1

Install NMIT before you install the SDK because the SDK installation procedure checks for the existence of an NMIT installation, and if one exists, proceeds to install the SDK into the <NMIT install directory>/MobileInternet/devices/Nokia subdirectory. This enables NMIT to locate the installation of the SDK and to add it to its list of available simulators.

If you install the SDK before installing NMIT, the SDK installation procedure does not find the NMIT installation and so does not install the SDK in the NMIT subdirectory. In this case, you can run the SDK standalone, but not with a subsequently installed NMIT.

When NMIT is started, it will detect that the SDK has been installed and will automatically add it to its menu of available devices. Then, when running NMIT, you can start the SDK by selecting it from NMIT's **Settings** menu.

Configuring NMIT

When NMIT has been installed prior to the SDK installation, then NMIT, when started, finds that the SDK (as well as any other Nokia mobile handset simulators) has been installed on the system. If the SDK has been installed, NMIT adds it to its list of available simulators, and you can then make the SDK the current simulator by selecting **Settings>Select Device>Nokia 3510i Content Authoring SDK**.

When the SDK is the currently selected simulator, you can choose the NMIT command **Settings>Device Settings** to display the **General** settings dialog box.

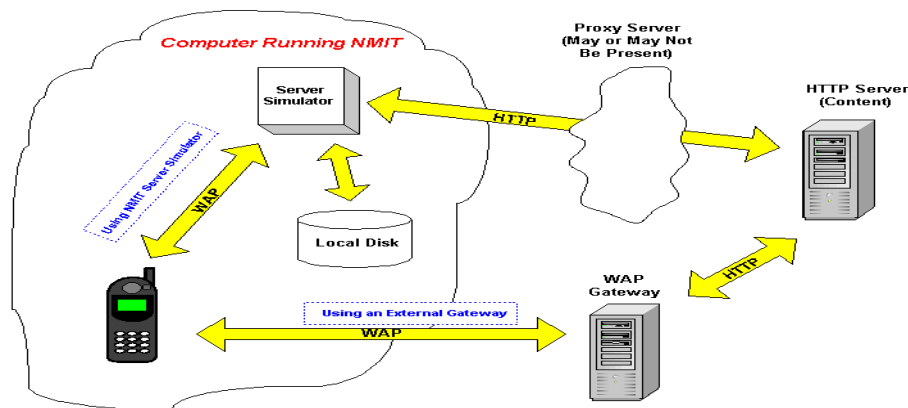
Configuring a Proxy Within NMIT

If you choose to use NMIT's server simulator as your WAP gateway, you may need to configure NMIT to use a proxy. To do this, enter the proxy name or its IP address within NMIT's **Connection** dialog (**Settings>Toolkit Preferences**).

You need to set a proxy address only if Internet requests made from your computer must pass through a proxy, which would be the case, for example, if you were working within a corporate Intranet. To clarify this, consider the following figure, which depicts a simulator within the NMIT environment.

You can see that when configured to use an external gateway, no proxy is needed because Internet requests go out over the WAP protocol, thus bypassing any corporate proxy server (typically an HTTP proxy server). On the other hand, when configured to use the server simulator, Internet requests leaving the server simulator are going out over the HTTP protocol and thus pass through any corporate proxy server.

If you configure the SDK to use an external WAP gateway other than server simulator, any proxy setting within NMIT is ignored.



Using Nokia 3510i Content Authoring SDK With Adobe GoLive

Adobe GoLive is available from Adobe Systems, Inc. Using GoLive with the SDK enables you to display on the SDK WML content that you create using GoLive's WML content editor.

Installing Adobe GoLive

You can install Adobe GoLive either before or after you install the SDK. Follow the installation instructions provided with the GoLive installation media; no special steps are required for use with the SDK.

You must install Nokia Developer's Suite for MMS (NDS for MMS), after you install Adobe GoLive. NDS for MMS is available for downloading at <http://www.forum.nokia.com/>. NDS for MMS 1.0 is automatically configured to run with the SDK. NDS for MMS beta 0.3 must be configured to run with the SDK.

Configuring Adobe GoLive for Previewing WML Content

By default, the WML module for content authoring is not accessible. To configure Adobe GoLive so that you can create a new WML file using a standard WML template and then display it on the SDK, perform the following steps:

1 Add the WML module.

Choose **Edit>Preferences** and then choose the **Modules** item. Navigate to the bottom of the list to the **Extend Scripts** folder and then place a check mark adjacent to the **WML** item. Finally, click **OK**.

Now you can create a new WML file by choosing **File>New Special>WML Deck**.

2 Add 3510i as a recognized browser.

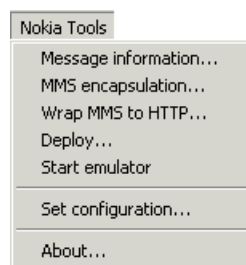
Choose **Edit>Preferences**, then **Browser**, and then **Add**. The **Choose Browser** dialog is displayed. Now navigate to the location of the SDK executable file (**3510i.exe**) and when its full path is displayed in the **Select items to add** text box, click **Done**. Now you can display a WML file within GoLive's content editor on the SDK by choosing **File>Preview In>3510i.exe**.

Configuring Adobe GoLive With NDS for MMS Beta 0.3

Nokia Development Suite for MMS 0.3 (NDS for MMS) is used to encode the SMIL file and the contents to which it refers and to add required MMS headers. This process is called “encapsulation” within NDS for MMS. Such encapsulated MMS messages can then be delivered and displayed on the SDK.

Install NDS for MMS only if you want to use the Adobe GoLive SMIL editor for creating MMS messages. Adobe’s SMIL editor provides facilities for specifying the media content files to be included in an MMS message and for specifying the presentation options for these media files.

You must install NDS for MMS after installing GoLive because the NDS for MMS installation adds a **Nokia Tools** menu item to the GoLive menu bar as shown below.



To install NDS for MMS, follow the installation instructions provided with the installation download.

Important: choose the **Integrate with Adobe GoLive** check box (not **Series 60 MMS SDK (standalone)**) in the third installation dialog.

NDS for MMS adds to Adobe GoLive the capability to display an MMS message on the SDK. To configure GoLive to display MMS messages on the SDK, choose **Nokia Tools>Set configuration**:

- 1 In the first text box, **Path to the emulator**, choose the **Browse** button and navigate to the location of the 3510i_GoLive.exe file.
- 2 In the second text box, **Path to the deployment folder**, choose the **Browse** button and navigate to the folder where the SDK was installed. This folder enables direct deployment to the SDK; that is, when you choose **Nokia Tools>Deploy**, the message is delivered immediately and displayed on the SDK. If you enter a different folder, direct deployment is disabled, and you will need to use the SDK’s **File>Open** menu item in order to display the message.

The SDK recognizes an incoming MMS message by inspecting the contents of the folder that you select here; the MMS message is not deployed over the network but simply placed into this folder. Do not modify the remaining three text entry boxes.

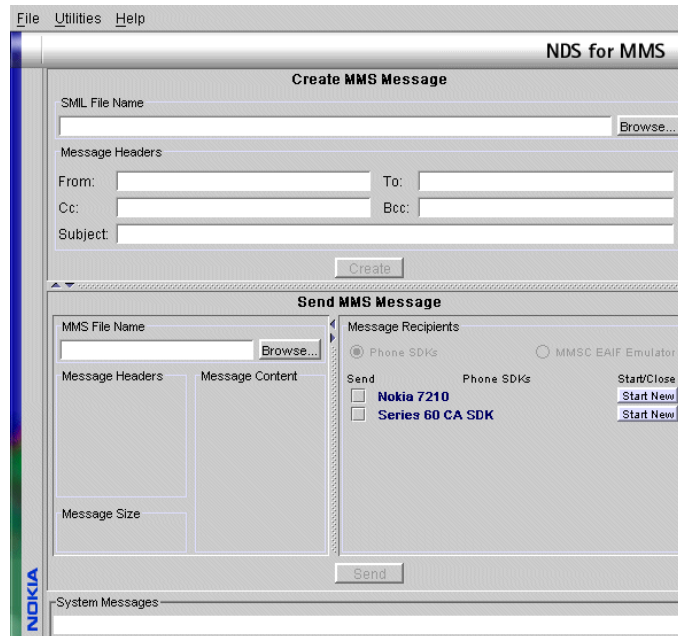
Configuring Adobe GoLive With NDS for MMS 1.0

Nokia Development Suite for MMS 1.0 (NDS for MMS) is automatically configured to run with the SDK within Adobe GoLive. You do not need to configure it manually.

You must install NDS for MMS after installing GoLive because the NDS for MMS installation adds a **Nokia Tools** menu to the GoLive menu bar. Under this menu, you'll see the option, **Create/Send MMS Messages**, which launches NDS for MMS 1.0 when you have a .SMIL file open.

To view MMS messages using NDS for MMS within Adobe GoLive:

- 1 Select **File>Open** and open a .SMIL file.
- 2 Select **Nokia Tools>Create/Send MMS Messages**.
NDS for MMS launches.



- 3 Fill in the **To** and **From** fields and click **Create**.
- 4 Select the SDK to which you will deploy the message.
- 5 Click **Send**.

Browsing, Receiving, and Creating Messages

This chapter covers how to browse, receive, and create messages using Nokia 3510i Content Authoring SDK and supporting applications.

About Browsable Content Types

Using the SDK's browser, you can download and display the following content types. The three columns to the right specify the mechanism for which the content type is supported:

File Type	MIME Type	Direct Download	Embedded in WML	Referenced by COD
Compiled WML	application/vnd.wap.wmlc	Yes	No	No
Compiled WMLScript	application/vnd.wap.wmlsc	No	Yes	No
vCard	text/x-vcard	Yes	No	Yes
vCalendar	text/x-vcal	Yes	No	Yes
OTT NRT	application/ vnd.nokia.ringing-tone	Yes	No	Yes
JPEG	image/jpeg	Yes	No	Yes
GIF87a GIF89a	image/gif	Yes	Yes	Yes
BMP	image/bmp	Yes	Yes	Yes
WBMP	image/vnd.wap.wbmp	No	Yes	No
PNG	image/png	Yes	No	No
MIDI	audio/midi, audio/sp-midi	No	No	No
JAD	txt/vnd.sun.j2me.app-descriptor	Yes	No	No
JAR	application/java	Yes	No	No
COD	text/x-co-desc	Yes	No	No

WML content coming into the SDK through a WAP gateway is already encoded. However, you can open local unencoded WML files: the SDK encodes this WML content before sending it to its phone software for display. If such WML files contain relative references to other supported content types, such as GIF images or WMLScripts, the SDK opens and displays these content types as well.

If WMLScript content is referenced from within a local WML file and the WMLScript file is not encoded, then the SDK encodes it before passing it to its phone software for processing.

By default, 3510i expects WML files to use the ISO-8859-1 (UTF-8) character set. The SDK cannot load WML files using other character sets unless the character set is specified in the WML file. You can specify an alternate character set using either (1) the `encoding` attribute within the `xml` declaration, or (2) the `charset` attribute within a `meta` element in the head section of the file.

Memory and Storage Constraints for Browsing

- Maximum WML deck size: 2.8KB
- Maximum URL length in requests: 538
- Maximum URL length as homepage: 100
- Maximum URL length as bookmark: 255
- Maximum URL length in a cached page: 128
- HTTP authentication: YES
- Maximum number of connection settings: 5
- Maximum length of a username: 32
- Maximum length of a password: 20

Browsing in the Standalone Nokia 3510i Content Authoring SDK

When you use by itself, use the **File>Open** menu item to display browsable content. To browse Internet content, the SDK must be configured to use a WAP gateway or NMIT's server simulator.

To browse...	Do the following...
Mobile Internet Content	Choose File>Open and then enter the desired URL into the Open dialog. See the section titled "Configuring Nokia 3510i Content Authoring SDK" if you experience problems.
Local File Content	Choose File>Open and then enter the full path of the desired file into the Open dialog, or choose the Browse button to navigate to the desired file.

Browsing Within NMIT

When using NMIT, you can browse both local files and mobile Internet content in several ways, as described in the following table:

To browse...	Do the following...
Mobile Internet Content	<ul style="list-style-type: none"> Choose the Browser>Load Location menu item and enter the desired URL in the dialog box. Choose the Bookmarks button and select a URL you have previously bookmarked. Enter the desired URL into the Go address bar and then click Return. Choose Browser>Home to display a previously defined (Settings>Toolkit Preferences) home page.
Local File Content	<ul style="list-style-type: none"> Choose the Browser>Load File menu item and enter the desired file path in the dialog box (or navigate to its location). Choose the Bookmarks button and select a file you have previously bookmarked. Enter the desired file path into the Go address bar and then click Return. Choose Browser>Home to display a previously defined (Settings>Toolkit Preferences) home page. Choose File>Open to open a WML file in NMIT's Editor window and then choose Show to encode and display the file. Choose File>New to create a new WML file in NMIT's Editor window and then choose Show to encode and display the file.

Browsing Within Adobe GoLive

You can browse local files only when you are using Adobe GoLive. You cannot browse mobile Internet content from within GoLive.

To browse...	Do the following...
Local File Content	<ul style="list-style-type: none"> Choose File>New Special>WML Deck to create a new WML file in GoLive's Editor window and then choose File>Preview In>3510i.exe to encode and display the file. Choose File>Open to open an existing WML file in GoLive's Editor window and then choose File>Preview In>3510i.exe to encode and display the file.

Receiving MMS Messages

The SDK accurately simulates the reception of an MMS message as it would be experienced by the user of a 3510i mobile handset. Unlike the handset, however, the SDK cannot send or forward an MMS message to another phone or to any other application.

Although it is possible to create an MMS message using the SDK's phone menus (as a user of a 3510i handset would do), it is not possible to save a message created in this way to a file. So you must use other applications to create MMS messages, which can then be received and displayed by the SDK.

The section titled [Creating MMS Messages](#) describes how to create MMS messages using either NMIT or Adobe GoLive. Such messages when saved to a file can be opened (received) by the SDK, used standalone, at any time.

No matter how an MMS message was created or how the SDK opens it, once it is received, the way the SDK displays it and the way a user of the SDK phone menus disposes of it are identical. The section titled [Receiving MMS Messages](#) describes the ways in which a user can deliver an MMS message to the SDK, and the section titled [Viewing and Disposing of an MMS Message](#) describes the phone menu options for disposing of MMS messages (saving, deleting, forwarding, and so on).

The following section, [Overview of MMS Messages](#), provides some relevant information about MMS message processing both in the real-world network environment and within the SDK simulation environment. (The WAP MMS Architecture Overview, available at <http://www.wapforum.org/>, discusses this subject in greater detail.)

Overview of MMS Messages

An MMS message contains rich media content that may include one or more text, image, or audio components. The message optionally can also include a SMIL (Synchronized Multimedia Integration Language) component that specifies how the client handset presents the parts of the message. The SMIL part specifies the layout, sequence, and timing of the parts in the MMS message.

An MMS message, which is encoded and includes headers for individual parts and for the message as a whole, is constructed from these parts:

- At the outermost (WSP) level, an MMS message consists of a **WSP Header** section and a WSP Content section. The `Content-Type` header specified within the WSP Header section must be `application/vnd.wap.mms-message`.
- The **WSP Content** section consists of a multipart message in WSP-encoded format consisting of an MMS Header section and an MMS Content section (also called “message body”).
- The **MMS Header** section must specify a `Content-Type` header of (1) `application/vnd.wap.multipart.related` if there is a SMIL file included (which defines or coordinates the presentation of the parts) or (2) `application/vnd.wap.multipart.mixed` if there is no included SMIL file.
- The **MMS Content** section (or message body) consists of each included part, where each part consists of a Part Header section and Part Content section. Parts are separated from one another by a boundary string defined in the MMS Header section.
- The **Part Header** section consists of a number of headers particular to the part.
- The **Part Content** section contains the actual part content.

The the SDK currently ignores any included SMIL part. Whether or not a SMIL part is included in the MMS message, the SDK displays the first media part. The user must then scroll to view any remaining media parts.

Content Types Supported Within an MMS Message

Nokia 3510i Content Authoring SDK can correctly handle an MMS message whose content consists of one or more parts of the following content types:

File Type	MIME Type
Text	text/plain
JPEG	image/jpeg
GIF87a and GIF89a	image/gif
BMP	image/bmp
PNG	image/png
WBMP	image/vnd.wap.wbmp
MIDI	audio/midi, audio/sp-midi, audio/mid, audio/x-midi
Nokia wallpaper	image/vnd.nok-wallpaper
Nokia operator logo	image/vnd.nok-oplogo
Nokia color operator logo	image/vnd.nok-oplogo-color
Nokia ringing tone	application/vnd.Nokia.ringing-tone

Memory and Storage Constraints for MMS Messages

Nokia 3510i Content Authoring SDK can correctly handle an MMS message of these sizes:

MMS message size	30 KB (maximum)
Wallpaper image	128x128 pixels (maximum)
Image content part in an MMS message	122x96 pixels (maximum)

MMS Messages in the Network and Simulation Environments

Nokia 3510i handset works within the Multimedia Messaging Service (MMS) application environment. This environment is a WAP-based, store-and-forward, rich-content messaging system. It uses the WAP WSP protocol stack for communication between a client device and a WAP gateway, and it uses a Multimedia Messaging Service Center (MMSC), which is a kind of combination MMS Proxy-Relay and MMS Server, as the store-and-forward center (similar to regular email).

Nokia 3510i Content Authoring SDK works within a simulation environment in which MMS messages are not delivered via the WAP WSP protocol between the client and WAP gateway. Instead, private interfaces are used to enable the SDK to receive MMS messages as if they had been delivered through the protocol stack. This occurs slightly differently depending on whether the MMS message is opened directly by the SDK, “pushed” from within NMIT’s MMS Message Editor, or previewed from within Adobe GoLive. Nevertheless, the resultant display of the MMS message on the SDK is identical to one in which the message displayed arrived via the protocol stack. In this respect, therefore, the SDK faithfully reproduces the experience of message delivery on a 3510i handset.

You can open local MMS files in the following three different ways:

- The GUI (see “Menu Commands”)
- The CLI (see “Command Line Interface”)
- Supporting Applications (see the section titled [Receiving MMS Messages](#) for a description of how NMIT and GoLive create and display MMS messages)

Creating MMS Messages

You can use either NMIT or Adobe GoLive to create MMS messages. These messages can be sent directly to the SDK display from within these supporting applications or saved as files with the `.mms` file extension (and these MMS files can be opened directly by the SDK).

Creating an MMS Message Using NMIT

When you create an MMS message using the NMIT MMS Message editor, you are creating a message that conforms to a structure that has a complicated format. For this reason, within NMIT, creating a new MMS message is a two-stage process.

When you choose **File>New>MMS Message**, NMIT opens the MMS Setup Wizard. Within the wizard’s dialogs, you enter information about recipients of the message, media files to be included in the message, and choose whether a SMIL file is to be used. The wizard facilitates creating an MMS message by supplying required headers and inserting default header values. Additionally, through the wizard you can choose to include an existing SMIL file, to specify no SMIL file, or to request NMIT to create a default SMIL file using the included media files you have specified.

After choosing **Finish** in the last wizard dialog, NMIT opens the standard MMS Message Multipart editor. Within this editor, you can edit the values of all headers, both for the entire MMS message and for the individual parts. Also, you can add additional, optional headers. Note that this detailed level of control over MMS message headers is a unique feature found only in NMIT, so if this level of control is important to you, you may want to use NMIT to create your MMS messages.

After creating the MMS message and checking that the SDK is the currently selected simulator, choose the **Push** button within the MMS Message Editor to “push” the MMS message to the SDK’s Message Inbox. NMIT inserts the WSP Header value `application/vnd.wap.mms-message` automatically, encodes the message, and pushes it to the SDK through a private interface.

NMIT automatically saves the MMS message to disk before pushing it to the SDK.

Creating an MMS Message Using Adobe GoLive

Adobe GoLive provides a different method than NMIT for creating MMS messages. You edit a SMIL document, entering the file paths for all media parts to be included in the MMS message. Then, you use the **Nokia Tools** menu to encode and “wrap” the MMS message in MMS headers, and then finally “deploy” the message to a folder. If the configured folder is ... \3510i\Inbox (see the section titled "Configuring Adobe GoLive With NDS for MMS Beta 0.3"), the message is immediately delivered to the SDK.

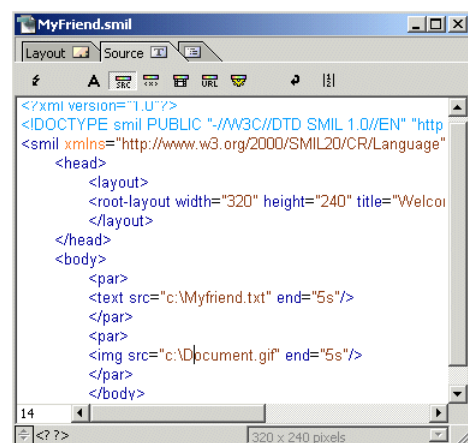
It is not necessary to choose the **Wrap MMS to HTTP** menu option before deploying the MMS message to the SDK. HTTP headers are required only when deploying an MMS message to a server.

To create an MMS message within Adobe GoLive:

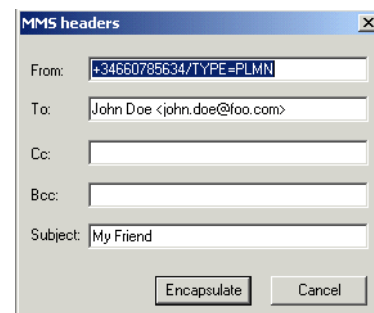
- 1 Choose **File>New Special>SMIL Document**. GoLive opens a SMIL document template in an editor window. The adjacent figure shows a SMIL document with two media files specified, a text document and a GIF image.

Tip: GoLive provides an Object Palette you can use when editing. Click the SMIL icon along the top of the Object Palette to display SMIL objects in the window region below. You can drag these objects into the editing window. When you drop them, they display as the related element tags. Now you can enter attribute names and values (such as file names of image, text, and sound files).

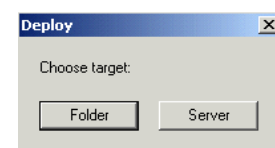
Tip: When you are finished entering data, check the validity of the document against the SMIL doctype by clicking the left-most icon in the toolbar at the top.



- 2 Choose **Nokia Tools>MMS encapsulation** to encode the content and add MMS headers for the header names shown in the dialog to the right. Header values are required for the **From** header, and for at least one of **To**, **CC**, or **BCC**. The **Subject** header is optional). Enter the desired values and then choose **Encapsulate**.



- 3 Choose **Nokia Tools>Deploy** and then choose the **Folder** button to write the MMS file to the deployment folder.



- 4 The message is immediately delivered to the SDK, as shown to the right, if you have configured ... \3510i\Inbox as your deployment folder. If you have chosen any other deployment folder, choose **File>Open**, navigate to the newly created MMS file, and then choose **OK**. The SDK then displays the message as shown.



Receiving MMS Messages

You can receive MMS messages through the SDK GUI, NMIT, and Adobe GoLive.

Receiving an MMS Message Using the SDK GUI

When using the SDK by itself, choose the **File>Open** menu item to open a local file having the file extension .mms.

The SDK announces that an MMS message has arrived (see right).

If the SDK is in browsing mode, the MMS message announcement is not displayed. You must use the phone menus to quit the browser in order to see the announcement.



Receiving an MMS Message Using NMIT

When using NMIT, use one of the following methods to deliver an MMS message to the SDK:

- Choose **File>Open** from NMIT's main menu or from the Toolkit Editors window, navigate to a local MMS file, and choose **Open**. NMIT displays the MMS file in the Toolkit Editors window. Choose **Push** to deliver that message to the SDK.
- Choose **File>New** from NMIT's main menu or from the Toolkit Editors window. Follow the procedure for creating a new MMS message described in the section titled [Creating an MMS Message Using NMIT](#). Choose **Push** to deliver that message to the SDK.

When you choose **Push**, the SDK announces that an MMS message has arrived (see right).

If the SDK is in browsing mode, the MMS message announcement is not displayed. You must use the phone menus to quit the browser in order to see the announcement.



Receiving an MMS Message Using Adobe GoLive

When using GoLive, use one of the following methods to deliver an MMS message to the SDK:

- Choose **File>Open** from GoLive's main menu, navigate to a local SMIL file, and perform steps 2 through 4 in the procedure described in the section titled [Creating an MMS Message Using Adobe GoLive](#).
- Choose **File>New** and perform all steps in the procedure described in the section titled [Creating an MMS Message Using Adobe GoLive](#).

After performing step 4 in the above procedure, the SDK announces that an MMS message has arrived (see right). If the SDK is in browsing mode, the MMS message announcement is not displayed. You must use the phone menus to quit the browser in order to see the announcement.



Viewing and Disposing of an MMS Message

In the following example, you see the reception and viewing of an MMS message in the SDK. This message consists of three media parts: The first part is a GIF image and the other two parts are text files.

When an MMS message has been received, the SDK displays the announcement message displayed to the right.



Choose **View** in the SDK (the left soft key) to display the first part contained in the MMS message. As shown in the figure to the right, this is a GIF image file.

If more than one MMS message has been delivered to the Message Inbox, pressing **View** results in a listing of the newly arrived MMS messages. You must then select the one you want to view. After this, choosing **View** displays the first part of the selected message.



Press the down arrow once to scroll to the next (and last) media part included in the MMS message. As shown in the figure to the right, this is a text file.

Pressing the scroll key again displays message header fields. Also note that the right soft key label has changed from **Back** to **Exit** because this is the last media part in the message. Pressing **Exit** causes the SDK to exit the Message Inbox and results in the display of the root screen (showing **Menu** and **Names** as the soft key labels).



Pressing the **Options** soft key displays the options for managing the message.

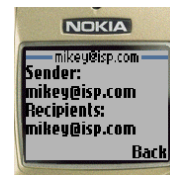
Choose **Delete message** to delete the entire MMS message.

Choose **Message details** to display information contained in the headers of the MMS message.

The options **Reply**, **Forward to no.**, and **Forward to e-mail** require a sending capability, which the SDK does not have. However, each of these options display dialogs faithful to those in the 3510i handset.

Pressing the **Options** soft key while viewing a multimedia part (an image or sound) displays a different set of options that provide for playing a sound part, saving an image to phone memory, and even using an image as wallpaper.

The figure to the right shows the message details associated with the MMS message.



Receiving Push Messages

You can use NMIT to deliver an SI (Service Indication) push message to the SDK's Service Inbox. Only Service Indication (SI) messages are supported.

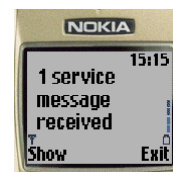
The SDK used alone, or used with Adobe GoLive, does not support push messages.

Push Message Content Type Support

An SI message is a text message sent to a client device for the purpose of notifying the user that a service or resource is available. The message contains a URL identifying this service. The SDK displays this text notification and you can choose to access the resource (the **Retrieve** option) or to delete the notification.

Viewing Push Messages Created in NMIT

To observe the text announcement that a message has been delivered to the SDK, the SDK must be in the state shown in the figure to the right. To do so, quit the SDK's browser. If the browser is active, the message is still delivered but you see no announcement of that fact until you quit the browser.



The following example procedure demonstrates how to push an SI message from within the NMIT SI editor and then use the menus within the SDK to view and process the received message.

This section describes how to use the menus within the SDK to process the received message. Using the keys on the SDK, you display an SI message as follows:

Within NMIT, push the SI message using either of the following procedures:

Choose **File>Open** from NMIT's main menu or from the Toolkit Editors window, navigate to a local SI file and choose **Open**. NMIT displays the SI file in the Toolkit Editors window. Choose **Push** to deliver that message to the SDK. Note that you may open a file with an extension `.si` (unencoded) or `.sic` (encoded); NMIT will encode the `.si` file before pushing it.

Choose **File>New** from NMIT's main menu or from the Toolkit Editors window. Enter the desired text for the text announcement in the **Text Message** box and the URL to be retrieved in the **href** box. Choose **Push** to deliver that message to the SDK.



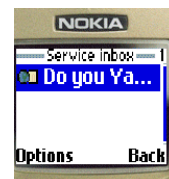
An SI message is sent to the SDK, having the content shown below:



The SDK acknowledges receipt of the message by displaying the text "1 service message received," as shown in the adjacent figure.

Choose **Show** in the SDK to open the Service Inbox.

The Service Inbox displays all available messages. As shown in the figure to the right, only one message is available.

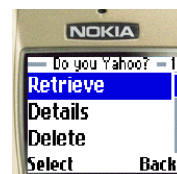


The most recent, unread messages always appear at the top. If you read a message but do not delete it, it is listed at the bottom.

Choose **Options** to display the options for managing the message.

The following options are displayed:

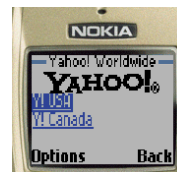
- **Retrieve.** Loads the URL specified by the `href` element in the Push message.
- **Details.** Displays as much of the text message as fits in the display (this is the title), the URL specified in the message (the value of the `href` element), and the creation and expiry dates.
- **Delete.** Deletes the message from the Service Inbox.



Choose **Retrieve** to load the URL. the SDK starts its browser and loads the specified URL as shown in the figure to the right.

In doing this, the SDK has activated its browser, so when you choose **Options**, the normal browser options are displayed.

If you choose **Back**, the query "Quit Browsing" is displayed. Choosing **Yes** here returns the SDK to the root menu structure.



Disposing of Push Messages

Unlike MMS messages which contain one or more parts of actual content, the main “content” of a Push message is the URL of the resource to be accessed. Therefore, there is no **Save** option for Push messages within the Service Inbox.

SI Push messages stay in the Service Inbox until you explicitly delete them.

Browser Features

This section covers the attributes of the browser features.

Rendering

- Plain text, links, input fields, and images on the same line: NO
- Horizontal alignment: YES
- Dynamic buttons: NO
- Tables: YES
- Image as a link: YES
- Wrapping modes: YES

Screen Attributes

- Screen size: 96x65
- Screen area reserved for content: 96x45
- Image formats: WBMP, GIF, Animated GIF
- Maximum number of rows of text: 4
- Maximum number of characters in a row: 11 to 47

HTTP Features Supported

- HTTP Post
- HTTP Cookies

WAP-Specific Functions Supported

- WML and WMLScript
- User Agent Profile (UAPROF)
- Wireless Telephony Application Interface (WTAI) functions, specifically, Make Call and Add Phone Book Entry. Note that, though these functions themselves are supported, it is not possible to make a phone call using the simulator.
- Secure connections using WTLS

Character Set Support

- ISO-8859-1
- UTF-8
- ISO-10646-UCS-2

Character encodings supported are fixed and cannot be changed.

Support for the above listed character sets does not imply support for the corresponding font glyphs, which would require both a localized version of Windows and the corresponding fonts.

Using Nokia 3510i Content Authoring SDK

This chapter is a reference for using the SDK graphical user interface (GUI), which closely replicates the user interface (UI) of the Nokia 3510i phone handset with respect to the following:

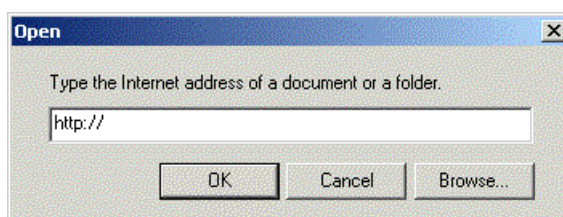
- General look
- Key layout
- Key definitions
- Error messages
- Vibration
- Display and rendering
- Soft keys
- Menu selection
- Backlight

Menu Commands

For standalone use as a browser and for configuration purposes, the SDK provides a menu bar consisting of File, Tools, and Help menus.

File Menu

Open...	Displays the following dialog:
----------------	--------------------------------



Enter directly the `file:///` URL or just the path of a `.wml` file and click **OK** or choose **Browse** to navigate to a local `.wml` file and click **Open** to display the file on the SDK.

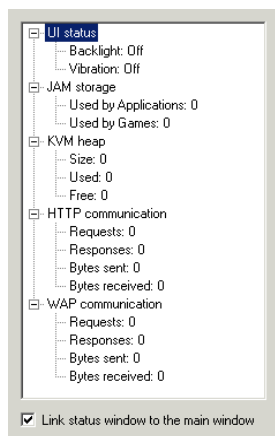
You can also specify an `http://` URL whose target is a `.wml` file, but this requires that you use a WAP gateway (or server simulator).

Exit	Closes the SDK application.
-------------	-----------------------------

Tools Menu

Status Window

Displays the **Status window**, shown below.



UI status

Indicates whether **Backlight** and **Vibration** are currently on or off in the SDK GUI.

JAM storage

Specifies the amount of heap memory currently being used to store MIDlets in the **Applications** and **Games** folders.

KVM heap

Specifies (1) the **Size** (in bytes) of the allocated KVM heap memory (you can set this value using the `-heapsize` command line parameter); (2) the memory currently being **Used**; and (3) the memory currently **Free** to be used.

HTTP Communication

Specifies the number of (1) **Requests** made, (2) **Responses** received, (3) **Bytes sent**, and (4) **Bytes received**.

WAP communication

Specifies the number of (1) **Requests** made, (2) **Responses** received, (3) **Bytes sent**, and (4) **Bytes received**.

These are all dynamic values, changing as you run and close MIDlets.

Link status window to the main window

When checked, the status window is linked to the main window so that it moves whenever the main window is moved; however, you can still move the status window independently of the main window.

When unchecked, each window can be independently positioned at any time.

Settings...

Displays the **Settings** dialog boxes that let you control how the SDK operates. The tab pages in **Settings** are described below.

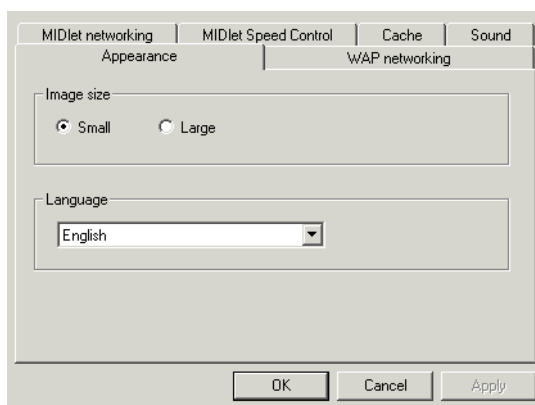


Image Size

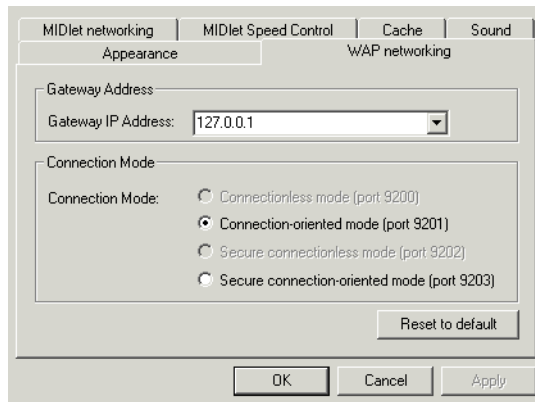
Choose which image size you prefer for the SDK. Both sizes function identically.

Language

Choose from the drop-down list box the language to be used when displaying the SDK's phone menus.

This setting affects only phone menus. Local files and Internet documents are displayed in the language used when the document was created.

You must restart the SDK after changing the language setting.



Gateway Address

Enter the IP address of the desired WAP Gateway in the text-entry box.

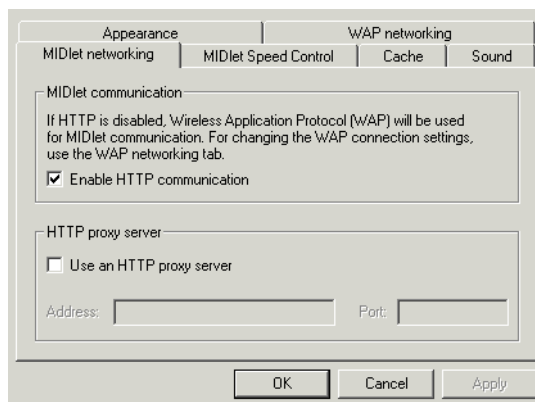
Note that if you have previously entered and saved an IP address in this dialog, you can recall it by choosing the arrow to the right of the box. This feature enables you to store the IP addresses of multiple WAP Gateways.

Connection Mode

Choose one of two connection modes. The port number for each is fixed and cannot be changed.

Note that the SDK stores the mode setting with the IP address specified in **Gateway Address**.

Choose **Reset to default** to restore the default Gateway Address (127.0.0.1) and Connection Mode (Connection-oriented mode).



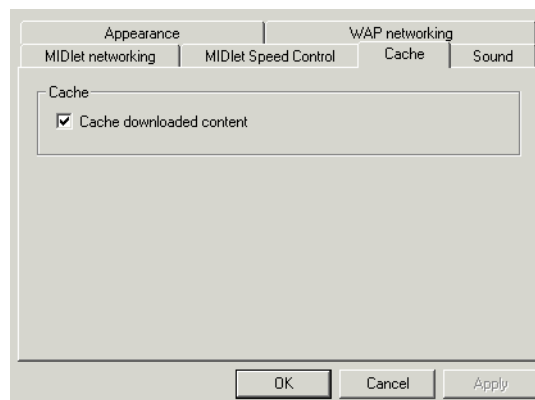
MIDlet communication

Check the **Enable HTTP communication** check box to specify that running MIDlets should use the HTTP protocol instead of WAP when making network requests.

If you uncheck this box, running MIDlets will use the WAP protocol over the WAP gateway specified in the **WAP networking** tab.

HTTP proxy server

Check the **Use an HTTP proxy server** check box if you have also enabled HTTP communication and your MIDlet is running behind an HTTP proxy server. In this case, also specify the address of the proxy server (for example, 113.14.176.11 or proxy.domain) and an (optional) port number. If the port number is not specified, the port 8080 is used.

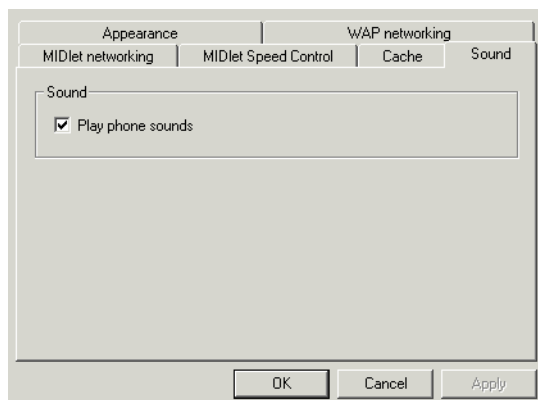


Cache

Choose **Cache downloaded content** to direct the SDK to store downloaded content in its cache.

If checked, every URL fetched is stored in cache. Then, whenever a URL request is made, the cache is checked first to see if the requested URL is stored there and if the content has not yet expired, and if so, it is used.

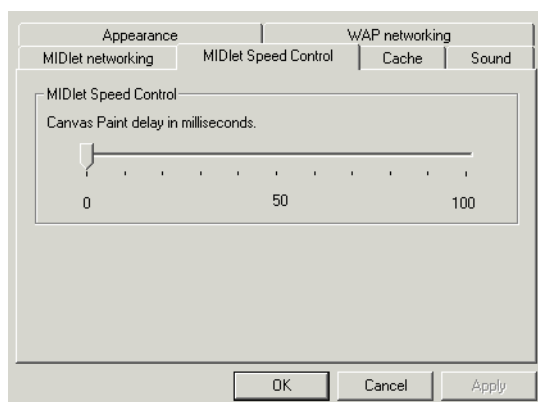
If not checked, each URL request is routed over the Internet, bypassing the cache.



Sound

Choose **Play phone sounds** to direct the SDK to send supported sound content to the computer's sound card.

Only sounds specified by a MIDlet are played.



MIDlet Speed Control

(Canvas Paint Delay Option)

This setting applies only to JAVA MIDlets. Use this setting to slow down MIDlets that display too rapidly on a fast computer.

Move the slider to the right to increase the delay in milliseconds of the canvas paint function. The canvas paint delay applies only to the SDK instance for which you set the delay. You must set the canvas paint delay for each new instance.

Because the maximum delay you can set is 100 milliseconds, this setting has no visible effect unless the canvas repaint rate of an object is significantly greater than 10 times per second.



Sound

Choose **Play phone sounds** to direct the SDK to send supported sound content to the computer's sound card.

Help Menu

About...

Displays the product name and version, as well as Nokia copyright information.

Phone Menus

The phone menu structure in the SDK GUI is identical to that of the 3510 phone handset, which is to say that no menus have been omitted from the SDK GUI. That said, however, not all menus in the SDK function similarly to their 3510 handset counterpart. This is because the SDK is not a phone but rather a tool for developers of Internet content. Nevertheless, to the extent possible, the SDK menus look and operate identically to 3510 handset menus even if the menu operation is not executed.

Running the SDK Through a Command Line Interface

Running the SDK from a command line interface lets you use batch files or create a shell in which you can repeatedly load commands to a single SDK instance.

To run the SDK from the Command Prompt, choose **Start menu>Run**, enter `cmd` in the **Run** dialog box, then type the executable in the command line.

Configuration options set on the command line are not persistent between instances of the SDK. If you need invocations to be persistent, use batch files. For example, a batch file containing the following line will cause the SDK to always start with the German language:

```
@3510i.exe -language german %*
```

You can run multiple instances of the SDK at the same time from the command line. Each instance is identified by a number enclosed in parentheses shown in the GUI title bar of each instance. If no instance exists when you deliver a CLI command, an instance is created. If an instance already exists, the command goes to that instance. If multiple instances are running, the command is directed to the highest numbered instance or to the instance you specify with the `-use` option.

Command Line Syntax

The syntax for starting the SDK from the command line is:

```
<full pathname>\3510i.exe [...-option] [parameter]
```

You can specify multiple options, but only one parameter per command line.

This table describes the components of the syntax:

<code>-option</code>	Options generally specify configuration settings for the SDK such as the language it should use in each SDK instance. Some options require a value. Option values must follow the option itself and be separated by a space. The SDK interprets all text preceded by a dash (-) as an option. You can specify multiple options, each one preceded by a dash, with a space between each option, as well as between an option and any subsequent or preceding option value. For example, the following command starts the SDK, specifying it should always remain on top of other windows and that it should display menu names in Spanish: <code>3510i.exe -ontop -language spanish</code>
----------------------	--

parameter You can have only one parameter per command line. Separate a parameter from any other element with spaces. The parameter and any options that go with it must be the last item on the line.

About Options

Options fall into two basic categories:

- Options that start the SDK, which include general and tracing options
- Options for Managing SDK Instances

Options for Starting the SDK

The following table lists command options that you can pass to the SDK when you start it. The options in this section, with the exception of `-help`, take effect only when the result of the command creates a new instance of the SDK. You can use the general command line options within the SDK standalone. The tracing command line options must be specified within an IDE development environment. Precede each option with a hyphen (-).

General Command Line Options

Option	Description	Default Value
<code>-help</code>	Specifies that the SDK display a window listing all available options with a brief description of each. The command line options described in this table is an expanded version of the help shown in the help window. If you have one or more instances running, use <code>-help</code> with <code>-new</code> . For example: <code>3510i.exe -new -help</code> . In this example, a new instance is created and a Help screen is displayed. When you click OK to close the Help screen, the instance also closes.	N/A
<code>-language</code>	Specifies the language that the SDK should use for its menu names upon startup. If the option value specified for the language option is unknown, the SDK ignores the option and starts up using the default language, which is the first language specified in the list of supported languages, usually English.	English
<code>-ontop</code>	Specifies that the SDK GUI should remain on top of all other open windows and must be minimized or moved to get at windows below it.	off
<code>-classpath</code>	Specifies the directories and <code>.jar</code> files to be searched for classes; the classpath may include both library and application class directories and <code>.jar</code> packages.	unspecified
<code>-heapsize</code>	Specifies the heap size (in bytes) available for MIDlets. Valid values range from 0 to 1,048,576 bytes (~1 MB). If the value specified is out of range, the SDK uses the default value (~200 KB).	204,796 bytes (~200 KB)
<code>-debugger</code>	Specifies that the MIDlet be run in debugging mode.	off
<code>-vm_port</code>	Specifies the TCP/IP port number between the emulator KVM and the debug proxy.	2800

General Command Line Options

Option	Description	Default Value
-parent_port	Specifies that the SDK, upon startup, is to connect to the local TCP port specified. This is used internally between SDK and some IDEs to inform the IDE that the SDK has exited. The SDK exits when it detects that the parent port has been closed; it also closes this socket if it exists.	unspecified
-protocol	Specifies the networking protocol for MIDlets. Valid values are either HTTP or WAP.	HTTP
-provisioning	Specifies that the SDK, upon startup, is to load the specified MIDlet from the hard drive using the MIDlet provisioning method rather than the immediate running method.	off
-http_proxy	Specifies the address and port number of the HTTP proxy, which may be needed for MIDlet HTTP networking. It is used only for the HTTP transport provided for MIDlets. The value is specified in the format address:port_number; for example, 113.14.176.11:8080 or proxy.domain:8080. If the port number is not specified, the port 8080 is used. By default, network requests made by MIDlets running in the SDK use the HTTP protocol. If the MIDlet you are testing is running behind an HTTP proxy, you must specify the address of this proxy; otherwise HTTP requests will fail.	unspecified
-wap_gateway	Specifies the IP address (and optional port number) for the gateway address that the SDK should use for its Internet connection upon startup. This argument requires at least one value (the IP address) but may be given an additional one, a port number. The port number, if specified, must be one of the two supported values (9201 and 9203) each of which is associated with a particular connection mode, as shown in the WAP networking dialog described in the section titled Tools Menu . For example, the following command starts the SDK, directing it to use the specified IP address in connection-oriented mode: 3510i.exe -wap_gateway 113.14.176.11:9201	127.0.0.1:921
-title	Replaces the number of the instance you specify with a name you specify. For example, you can rename the window (1) 3510i SDK to TEST 3510i SDK by entering 3510i.exe -title TEST. This option does not affect the -qinst option, which always lists the instances by their numbers or the -use option, which requires that you specify the instance by its original number.	N/A
-inbox	Specifies the directory that the new instance of the SDK checks for MMS files. For example: 3510i.exe d:\MMSFiles. When you place MMS files in the directory, the SDK displays and then deletes them. The directory name can be absolute or relative to the location of 3510i.exe. If the directory does not exist when you use the -inbox option, the SDK creates it. This function is available only through the command line interface.	N/A

Tracing output can be displayed only within the output window of an IDE, so you cannot capture tracing output when you run the SDK as a standalone application or with only NDS for J2ME without an IDE. Tracing output can be voluminous.

Tracing Command Line Options	Description	Default Value
-tracememoryallocation	Memory allocation trace	off
-tracegc	Garbage collection trace	off
-tracegcverbose	Verbose garbage collection trace	off
-traceclassloading	Class loading trace	off
-traceclassloadingverbose	Verbose class loading trace	off
-traceverifier	Verifier trace	off
-tracestackmaps	Stackmap trace	off
-tracebytecodes	Byte code trace	off
-tracemethodcalls	Method trace call	off
-tracemethodcallsverbose	Verbose method call trace	off
-tracestackchunks	Stackchunk trace	off
-traceframes	Frame trace	off
-traceexceptions	Exception trace	off
-traceevents	Event trace	off
-tracemonitors	Monitor trace	off
-tracethreading	Threading trace	off
-tracenetworking	Networking trace	off
-traceall	Enable all Java traces	off

Options for Managing SDK Instances

This section describes command line options you can use to manage SDK instances.

Option	Description	Values
-new	Creates a new SDK instance.	N/A
-use:	Directs a parameter to the existing instance you specify. Use the number originally issued to the instance, not the name issued with the -title option. For example: 3510i.exe -use:3	N/A
-shell	Creates a shell in which you can repeatedly load commands to the SDK instance within the shell.	N/A
-qinst	Displays the numbers of all running instances. This option always returns the instance numbers, even if you rename an instance with -title. Use -qinst as the sole option in a command line.	N/A

About Parameters

The following table lists the parameters you can use on the command line:

Command Line Parameters	Description	Values/Options
load	<p>Specifies a filename (.mms or .wml) or a URL string to be loaded into an instance. For example:</p> <ul style="list-style-type: none"> To load a local .wml file: 3510i.exe load C:\welcome.wml To load a remote .wml file: 3510i.exe load http://server.domain/welcome.wml To imply the load parameter: 3510i.exe http://server.domain/welcome.wml 	<ul style="list-style-type: none"> .mms filename .wml filename URL string
close	<p>The close parameter works in two ways, depending whether you are within a shell or outside a shell (see "Working With a CLI Shell"):</p> <ul style="list-style-type: none"> When you use close within a shell, close terminates the shell, but not the instance running in the shell. To also terminate the instance, you must use close with -exit. Close then terminates the shell and -exit terminates the instance within the shell. When you use close outside a shell, the parameter does nothing unless you use it with either -exit (to terminate the highest numbered SDK instance or the instance you specify with -use) or -exitall (to terminate all open SDK instances). <p>The options -exit and -exitall work only with the close parameter and, when used, must immediately follow the parameter.</p>	<p>-Exit. Closes one of the following:</p> <ul style="list-style-type: none"> Within a shell: the SDK instance within the shell. Outside a shell: the SDK instance you specify with -use or the SDK instance with the highest number. <p>-Exitall. Used only with the close parameter outside a shell to close all running SDK instance.</p>

You can enter only one parameter on the command line. The parameter and any options that go with it must be the last item on the line. For example:

- 3510i.exe load http://server.domain/welcome.wml
This example loads the site http://server.domain/welcome.wml into the SDK instance that is created.
- 3510i.exe close -exitall
This example closes all open SDK instances.

Working With a CLI Shell

Use the -shell option to create a session where you can repeatedly send commands (parameters) to the one instance without having to recreate the instance with every parameter. You can attach a shell to an instance that was created through a standalone SDK or through an application such as JBuilder. In this example, more than one instance is running:

```

3510i.exe -shell -use:2  ← Create a shell using instance 2.

%%-->> instance : 2

-> load http://server.domain/
    welcome.wml  ← Load this web site into instance 2 in the shell.

%%-->> load : ok

-> load c:\welcome.wml  ← Load this .wml file into instance 2 in the shell.

%%-->> load : ok

-> close  ← Close the shell. Instance 2 continues to exist.

%%-->> detaching

```

To close both the shell and the instance within the shell, use the `close` parameter with the `-exit` option:

```

-> close -exit
%%-->> exiting

```

In the example above, the `close` parameter closes the session, and the `-exit` option closes the instance within the session.

Keyboard Shortcuts

The table below shows each keyboard key or key combination that performs an action on the SDK. You may use these keyboard keys or key combinations instead of using the mouse to press keys on the simulator.

Keyboard Mapping to the SDK Keys

Key	Keyboard Key or Key Combination
Left soft key	F9 F11 Enter
Right soft key	F12 Backspace
Cursor Up	<Up arrow>
Cursor Down	<Down arrow>
Send key	NumPad / (slash)
End key.	NumPad - (minus)
# (pound) key	NumPad . (period)
* (asterisk) key.	NumPad *

Keyboard Mapping to the SDK Keys

Key	Keyboard Key or Key Combination
Number keys 0 through 9.	NumPad 0 through NumPad 9
Power On/Off	Not mapped
Volume Up	Not mapped
Volume Down	Not mapped

QWERTY Text Input

The SDK provides the ability for you to type on the QWERTY keyboard of your computer and have the characters typed appear in the display screen of the SDK. This feature is termed QWERTY text input.

For example, if the SDK browser is currently displaying a text entry box, say because a `wml` page you browsed to contained this box, you can enter characters into the box by typing on the QWERTY keyboard.

Or, to take another example, if the browser is not active at all, you can type on the QWERTY keyboard and see characters displayed on the SDK. The effect of this is quite similar to simply pressing the alphanumeric keys on a 3510i mobile handset when its browser is not active, as if you were entering a phone number to be dialed.

QWERTY text input is provided as an ease of use feature. However, it is important to note that characters entered into the SDK in this manner are not mapped directly to the SDK keys, as are the shortcut keys listed in the section titled [Keyboard Shortcuts](#).

To clarify this point, if you press the shortcut key *Numpad 5*, you will see the Number 5 key on the SDK flash briefly to a lighter hue, indicating a key press on the SDK GUI. In contrast, this does not happen if you press the 5 key on the QWERTY keyboard because this 5 key is not mapped directly to an SDK key.

